AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application:

LISTING OF CLAIMS:

1. (Currently Amended) A cryptographic method during which an integer division of the type q = a div b and r = a mod b is performed in a processor of an electronic device, where q is a quotient, a is a number containing m bits, b is a number containing n bits, with n less than or equal to m and b_{n-1} is non-zero, b_{n-1} being the most significant bit of b, comprising the following steps:

(i) performing a partial division of a word A, comprising n bits of the number a, by the number b to obtain a bit of the quotient q, wherein at least one of the numbers a and b comprises secret data;

(ii) repeating step (i) for m-n+1 iterations with the same number and type of operations being performed at each iteration, regardless of the value of the quotient bit obtained, to obtain the quotient q; and

(iii) generating encrypted or decrypted data in accordance with said quotient.

2. (Previously Presented) A method according to Claim 1, wherein, at each iteration, an addition of the number b to the word A and a subtraction of the number b from the word A are performed.

3. (Currently Amended) A method according to claim 1, wherein all the following steps are performed :

Input : $a = (0, a_{m-1}, \ldots, a_0)$

$b = (b_{n-1}, \ldots, b_0)$

Output: q = a div b and r = a mod b

$A = (0, a_{m-1}, \ldots, a_{m-n+1})$; σ' <- 1

For j = 1 to (m-n+1), do:

a <- $SHL_{m+1}(a, 1)$; σ <- carry

A <- $(\sigma')SUB_n(A, b) + (\neg\sigma')ADD_n(A, b)$

σ <-(σ' AND σ') / (σ' AND carry)/ (σ' AND carry)

lsb(a) σ'

σ' <- σ

End For

if ($\neg\sigma$ = TRUE) then A <- $ADD_n(A, b)$

wherein the symbol <- indicates loading of a content of a register containing data on the right of the symbol in a register whose data has the label on the left of the symbol;

wherein σ indicates whether or not a subtraction has been performed wrongly;

wherein $\neg\sigma$ is a negation of σ;

wherein σ' is a variable to preserve the value of σ obtained in a previous iteration;

wherein TRUE is a constant;

wherein lsb(a) is the lowest weight bit of the number a;

wherein $SHL_{m+1}(a, 1)$ is an operation of shifting to the left by 1 bit in the register of m+1 bits containing the data item a, the bit leaving the register being

stored in the variable carry and a bit equal to 0 being entered as the least significant bit of the register initially containing the data a;

wherein $ADD_n(A, b)$ is an operation of addition of the n bits of the number b to the n bits of the word A; and

wherein $SUB_n(A, b)$ is an operation of subtraction of the number b from the word A.

4. (Currently Amended) A method according to Claim 1 wherein, at each iteration, either the number b or ~~of~~ a number $\bar{b}$ complementary to the number b is added to the word A.

5. (Previously Presented) A method according to Claim 4, further including the step, at each iteration, of updating a first variable (σ') indicating whether, during the following iteration, the number b or the number $\bar{b}$ is to be added with the word A according to the quotient bit produced.

6. (Currently Amended) A method according to Claim 4, wherein all the following steps are performed :

Input : $a = (0, a_{m-1}, \ldots, a_0)$

$b = (b_{n-1}, \ldots, b_0)$

Output: $q = a \text{ div } b$ and $r = a \bmod b$

$A = (0, a_{m-1}, \ldots, a_{m-n+1})$; $\sigma' \leftarrow 1$; $\bar{b} \leftarrow CPL2_N(b)$

For j = 1 to (m-n+1), do:

$a \leftarrow SHL_{m+1}(a, 1)$; $\sigma \leftarrow carry$

d_{addr} <- b_{addr} + σ' ($\bar{b}_{addr}$ – b_{addr})

A <- ADD_n(A, d)

σ <-(σ' AND σ') / (σ' AND carry)/ (σ' AND carry)

lsb(a) <- σ'

σ' <- σ

End For

if (¬σ = TRUE) then A <- ADD_n(A, b);

wherein the symbol <- indicates loading of a content of a register containing data on the right of the symbol in a register containing data on the left of the symbol;

wherein σ indicates whether or not a subtraction has been performed wrongly;

wherein ¬σ is a negation of σ;

wherein σ' is a variable to preserve the value of σ obtained in a previous iteration;

wherein TRUE is a constant;

wherein lsb(a) is the lowest weight bit of the number a;

wherein SHL_{m+1}(a, 1) is an operation of shifting to the left by 1 bit in the register of m+1 bits containing the data item a, the bit leaving the register being stored in the variable carry and a bit equal to 0 being entered as the least significant bit of the register initially containing the data a;

wherein ADD_n(A, b) is an operation of addition of the n bits of the number b to the n bits of the word A;

wherein addr denotes address of a variable; and

wherein complement to 2^n of a number is obtained by the $CPL2_N$ of the number.

7. (Previously Presented) A method according to Claim 1, further including the steps, at each iteration, of performing an operation of complement to 2^n of an updated data item (b or $\bar{b}$) or of a notional data item (c or $\bar{c}$), and adding the updated data item with the word A.

8. (Previously Presented) A method according to Claim 7, further including the step, at each iteration, of updating a second variable (δ), indicating whether, during the following iteration, the operation of complement to 2^n is to be performed on the updated data item or on the notional data item.

9. (Previously Presented) A method according to claim 7, further including the step, at each iteration, of updating a third variable (β) indicating whether the updated data item is equal to the data item b or to its complement to 2^n.

10. (Currently Amended) A method according to claim 7, wherein all the following steps are also performed :

Input : $a = (0, a_{m-1}, \ldots, a_0)$

$b = (b_{n-1}, \ldots, b_0)$

Output: q = a div b and r = a mod b

$\sigma' \leftarrow 1$; $\beta \leftarrow 1$, $\gamma \leftarrow 1$; $A = (0, a_{m-1}, \ldots, a_{m-n+1})$

for j = 1 to (m-n+1), do:

$a \leftarrow SHL_{m+1}(a, 1)$; $\sigma \leftarrow$ carry

$\delta \leftarrow \sigma' / \beta$

d_{addr} <- b_{addr} + δ (c_{addr} – b_{addr})

d <- $CPL2_n$(d)

A <- ADD_n(A, b)

σ <-(σ AND σ') / (σ AND carry)/ (σ' AND carry)

β <- $\neg\sigma'$; γ <- γ / δ; σ' <- σ

lsb(a) = σ

end for

if ($\neg\sigma$ = TRUE) then A <- ADD_n(A, b);

wherein the symbol <- indicates loading of a content of a register containing data on the right of the symbol in a register containing data on the left of the symbol;

wherein σ indicates whether or not a subtraction has been performed wrongly;

wherein $\neg\sigma$ is a negation of σ;

wherein σ' is a variable to preserve the value of σ obtained in a previous iteration;

wherein TRUE is a constant;

wherein lsb(a) is the lowest weight bit of the number a;

wherein SHL_{m+1}(a, 1) is an operation of shifting to the left by 1 bit in the register of m+1 bits containing the data item a, the bit leaving the register being stored in the variable carry and a bit equal to 0 being entered as the least significant bit of the register initially containing the data a;

wherein ADD_n(A, b) is an operation of addition of the n bits of the number b to the n bits of the word A;

wherein addr denotes address of a variable; and

wherein β and γ are variables.

11. (Currently Amended) A method according to Claim 10, wherein, at the end, the following operations are performed :

if ($\neg\beta$ = TRUE) then b <- $CPL2_n(b)$

if ($\neg\gamma$ = TRUE) then c <- $CPL2_n(c)$;

wherein $\neg\beta$ is a negation of β; and

wherein $\neg\gamma$ is a negation of γ.

12. (Previously Presented) An electronic component comprising calculation means programmed to implement a method according to claim 1, said calculation means comprising a central unit associated with a memory comprising several registers for storing the data a and b.

13. (Previously Presented) A chip card comprising an electronic component according to Claim 12.